CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

UMR 6057 CNRS
PAROLE ET LANGAGE

Université de Provence - 29, Avenue Robert Schuman 13
621 Aix-en-Provence Cedex 01- France

Format de fichiers R.I.F.F (Wave, Sesane...)

Document	TN Soft-Format Fichier SESANE EVA Wsig.doc			Version	1	Révision	0
Auteur	A. Ghio			Revu		Approuvé	
Créé le	29/04/08	Imprimé le	07/05/08	Nb pages	16		

Ce document a pour but de recueillir les informations techniques concernant le format de fichiers RIFF (Resource Interchange File Format) dont sont issus les fichiers Wave et SESANE (EVA, DIANA).

Table des Matières

Terminologie

SESANE est l'environnement logiciel commun aux appareils d'analyse EVA et DIANA. Les fichiers SESANE sont donc identiques pour les deux plateformes.

Les fichiers obtenus avec SESANE sont des fichiers RIFF.

Dans le cadre des ressources multimédia, le format RIFF (Resource Interchange File Format) est incontournable, que ce soit pour l'audio ou la vidéo. À proprement parler, RIFF n'est pas un format unique de fichier, mais un principe d'architecture de fichier. De tels fichiers sont composés de différents blocs (appelés Chunks), certains blocs définissant des caractéristiques techniques (nb de canaux, fréquence d'échantillonnage…), des méta-données (commentaires, contenu, copyright …) et contenant les données proprement dites… Parmi les fichiers RIFF, on peut citer le format Wave pour l'audio, l'AVI pour la vidéo. Il faut signaler que même pour un format précis (ex : wav), il existe plus d'une trentaine de sous type (ex : wav-PCM, ADPCM, u-law)…. De même, l'AVI video peut être de sous-type DivX, DV, Cinepak, Indeo…
Dans des applications multiparamétriques (ex : fichiers simultanés audio, aéro, EGG avec EVA), nous fournissons les données dans un format RIFF particulier de façon à inclure des informations de calibrage (ex : valeurs en l/s pour les débits, hPa pour pressions) et autres méta-données.

Convention sur les variables

BYTE :	1octet	8 bits
WORD ⇔ WORD16 :	2 octets non signés	16 bits
INT16 :	2 octets signés	16 bits
DWORD ⇔ WORD32 :	4 octets	32 bits
INT32 :	4 octets signés	32 bits
FOURCC⇔char[4]	4 octets	32 bits

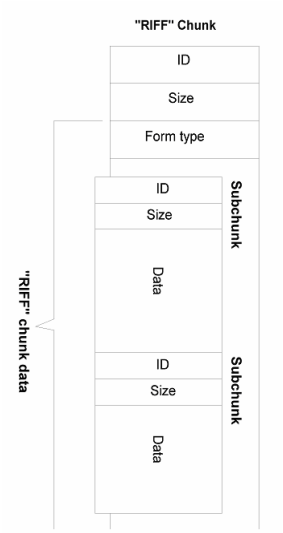
La définition de type FOURCC signifie Four Character Code ⇔ séquence de 4 caractères ASCII permettant de fournir un identifiant (ex : 'R''I''F''F', 'W''A''V''E', 'd''a''t''a', 'L''I''S''T')

Structure générales des fichiers RIFF

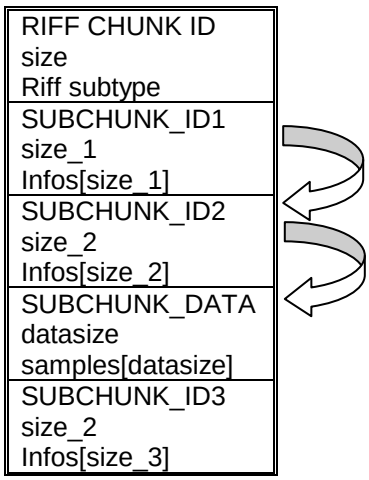
La structure de base des fichiers RIFF est appelé chunk. Chaque chunk est constitué des éléments suivants :

```
typedef struct  {
        FOURCC      ckID ;          // code sur 4 caractères spécifiant l'identité
        DWORD       ckSize ;        // taille des données qui suivent
        BYTE        ckData[ ckSize ];       // tableau des données
} CK ;
```

Le premier chunk d'un fichier RIFF doit avoir un identificateur "RIFF"[1].
Tous les autres chunks du fichier sont des sous-chunks (subchunks) du chunk RIFF.
Les seuls chunks autorisés à contenir des sous-chunks sont ceux dont l'identificateur est "RIFF" ou "LIST".

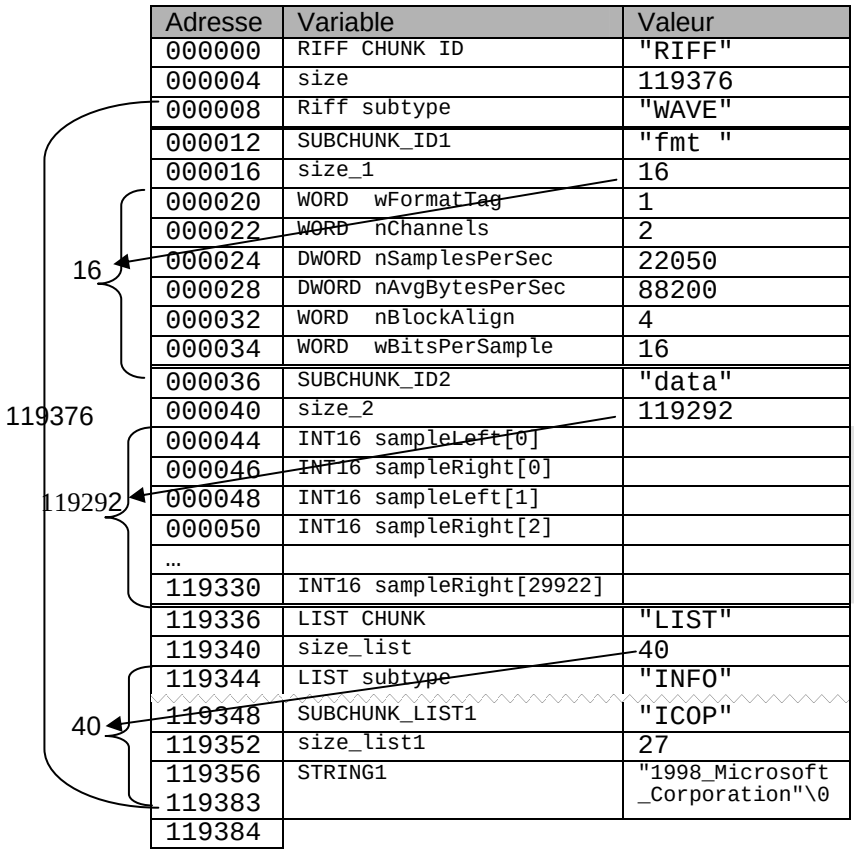


L'intérêt de cette structure est la possibilité d'identifier des blocs de données (par l'identificateur FOURCC), de les lire ou de sauter des informations optionnelles, la taille des blocs étant fournie par ckSize.



[1] Tout fichier RIFF commence par les 4 caractères "RIFF". Cela permet de vérifier et contrôler immédiatement si le fichier à lire est bien au format RIFF.

Exemple pour fichier Wave (C:\WINDOWS\Media\notify.wav)



Adresse	Variable	Valeur	
000000	RIFF CHUNK ID	"RIFF"	Systématique pour fichier RIFF[1]
000004	size	119376	
000008	Riff subtype	"WAVE"	Ce fichier RIFF est un ficher Wave.
000012	SUBCHUNK_ID1	"fmt "	Identificateur du format Wav
000016	size_1	16	Le header WAVEFORMAT fait 16 octets.
000020	WORD wFormatTag	1	=> Wave PCM
000022	WORD nChannels	2	=> stéréo
000024	DWORD nSamplesPerSec	22050	=> 22050 Hz
000028	DWORD nAvgBytesPerSec	88200	=> 88200 octets/sec = 22050*2(16bits)*2(stéréo)
000032	WORD nBlockAlign	4	=> alignement sur 4 octets (16 bits stéreo)
000034	WORD wBitsPerSample	16	=> 16 bits
000036	SUBCHUNK_ID2	"data"	Identificateur des données sonores
000040	size_2	119292	
000044	INT16 sampleLeft[0]		DONNEES SONORES
000046	INT16 sampleRight[0]		
000048	INT16 sampleLeft[1]		
000050	INT16 sampleRight[2]		
…			
119330	INT16 sampleRight[29922]		
119336	LIST CHUNK	"LIST"	Chunk List = informations textuelles
119340	size_list	40	
119344	LIST subtype	"INFO"	
119348	SUBCHUNK_LIST1	"ICOP"	Information Copyright
119352	size_list1	27	Taille de la chaîne de caractères
119356	STRING1	"1998_Microsoft_Corporation"\0	chaîne de caractères
119383			
119384			

Le chunk "RIFF"

Identificateur (ckID)

L'identificateur est toujours égal à ckID = "RIFF"

Taille (ckSize)

La taille ckSize est égale à la taille des données qui suivent.
On peut noter que :

ckSize = taille du fichier - sizeof(FOURCC) - sizeof(DWORD) = taille du fichier - 8

Taille de ckID

Taille de ckSize

Données (ckData)

Form type

Le chunk "RIFF" comprend une zone obligatoire de 4 octets en début de tableau des données. Cette zone fournit un identificateur de type (form type) au format FOURCC. Cet identificateur indique le type de fichier RIFF. Par exemple,

- un fichier « Microsoft waveform audio files» a un form type = "WAVE"
- un fichier « SESANE » a un form type = "WSIG"

Les sous-chunks de description

Les subchunks incorporent les informations nécessaires à l'exploitation des données proprement dites.
Ils sont de la forme :

	ckID	ckSize	ckData
	FOURCC (4 octets)	DWORD	BYTE[ckSize]
ex : pour wave	"fmt "	sizeof(struct WAVEFORMATEX)	struct WAVEFORMATEX
ex : pour SESANE	"sdsc"	sizeof(struct SIG_DESC)	struct SIG_DESC
	"adsc"	sizeof(struct ACQ_DESC)	struct ACQ_DESC

Le sous-chunk de données

Le subchunk "data" incorpore les données proprement dites. Il est toujours de la forme :

ckID	ckSize	ckData
FOURCC (4 octets)	DWORD	BYTE[ckSize]
"data"	taille des données en octets	données

Le chunk "fmt " pour fichiers wave

The WAVEFORMATEX structure defines the format of waveform-audio data. Only format information common to all waveform-audio data formats is included in this structure. For formats that require additional information, this structure is included as the first member in another structure, along with the additional information.

```
typedef struct {
    WORD  wFormatTag;
    WORD  nChannels;
    DWORD nSamplesPerSec;
    DWORD nAvgBytesPerSec;
    WORD  nBlockAlign;
    WORD  wBitsPerSample;
    WORD  cbSize;
} WAVEFORMATEX;
```

Variables members du chunk "fmt " d'un fichier wave

wFormatTag
: Waveform-audio format type. Format tags are registered with Microsoft Corporation for many compression algorithms. A complete list of format tags can be found in the MMREG.H header file[*] .

nChannels
: Number of channels in the waveform-audio data. Monaural data uses one channel and stereo data uses two channels.

nSamplesPerSec
: Sample rate, in samples per second (hertz), that each channel should be played or recorded. If wFormatTag is WAVE_FORMAT_PCM, then common values for nSamplesPerSec are 8.0 kHz, 11.025 kHz, 22.05 kHz, and 44.1 kHz. For non-PCM formats, this member must be computed according to the manufacturer's specification of the format tag.

nAvgBytesPerSec
: Required average data-transfer rate, in bytes per second, for the format tag. If wFormatTag is WAVE_FORMAT_PCM, nAvgBytesPerSec should be equal to the product of nSamplesPerSec and nBlockAlign. For non-PCM formats, this member must be computed according to the manufacturer's specification of the format tag.
: Playback and record software can estimate buffer sizes by using the nAvgBytesPerSec member.

nBlockAlign
: Block alignment, in bytes. The block alignment is the minimum atomic unit of data for the wFormatTag format type. If wFormatTag is WAVE_FORMAT_PCM, nBlockAlign should be equal to the product of nChannels and wBitsPerSample divided by 8 (bits per byte). For non-PCM formats, this member must be computed according to the manufacturer's specification of the format tag.
: Playback and record software must process a multiple of nBlockAlign bytes of data at a time. Data written and read from a device must always start at the beginning of a block. For example, it is illegal to start playback of PCM data in the middle of a sample (that is, on a non-block-aligned boundary).

wBitsPerSample
: Bits per sample for the wFormatTag format type. If wFormatTag is WAVE_FORMAT_PCM, then wBitsPerSample should be equal to 8 or 16. For non-PCM formats, this member must be set according to the manufacturer's specification of the format tag. Note that some compression schemes cannot define a value for wBitsPerSample, so this member can be zero.

cbSize
: Size, in bytes, of extra format information appended to the end of the WAVEFORMATEX structure. This information can be used by non-PCM formats to store extra attributes for the wFormatTag. If no extra information is required by the wFormatTag, this member must be set to zero. Note that for WAVE_FORMAT_PCM formats (and only WAVE_FORMAT_PCM formats), this member is ignored.

Remarks
: An example of a format that uses extra information is the Microsoft Adaptive Delta Pulse Code Modulation (MS-ADPCM) format. The wFormatTag for MS-ADPCM is WAVE_FORMAT_ADPCM. The cbSize member will typically be set to 32. The extra information stored for WAVE_FORMAT_ADPCM is coefficient pairs required for encoding and decoding the waveform-audio data.

[*] voir plus loin

Liste des FormatTag d'un fichier wave (PCM, ADPCM, A-Law…)

```
/* WAVE form wFormatTag IDs */

#define  WAVE_FORMAT_UNKNOWN    0x0000  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_PCM  1
#define  WAVE_FORMAT_ADPCM      0x0002  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_IEEE_FLOAT 0x0003  /*  Microsoft Corporation  */
                                        /*  IEEE754: range (+1, -1]  */
                                        /*  32-bit/64-bit format as defined by */
                                        /*  MSVC++ float/double type */
#define  WAVE_FORMAT_IBM_CVSD   0x0005  /*  IBM Corporation  */
#define  WAVE_FORMAT_ALAW       0x0006  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_MULAW      0x0007  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_OKI_ADPCM  0x0010  /*  OKI  */
#define  WAVE_FORMAT_DVI_ADPCM  0x0011  /*  Intel Corporation  */
#define  WAVE_FORMAT_IMA_ADPCM  (WAVE_FORMAT_DVI_ADPCM) /*  Intel Corporation  */
#define  WAVE_FORMAT_MEDIASPACE_ADPCM   0x0012  /*  Videologic  */
#define  WAVE_FORMAT_SIERRA_ADPCM       0x0013  /*  Sierra Semiconductor Corp  */
#define  WAVE_FORMAT_G723_ADPCM 0x0014  /*  Antex Electronics Corporation  */
#define  WAVE_FORMAT_DIGISTD    0x0015  /*  DSP Solutions, Inc.  */
#define  WAVE_FORMAT_DIGIFIX    0x0016  /*  DSP Solutions, Inc.  */
#define  WAVE_FORMAT_DIALOGIC_OKI_ADPCM 0x0017  /*  Dialogic Corporation  */
#define  WAVE_FORMAT_MEDIAVISION_ADPCM  0x0018  /*  Media Vision, Inc. */
#define  WAVE_FORMAT_YAMAHA_ADPCM       0x0020  /*  Yamaha Corporation of America  */
#define  WAVE_FORMAT_SONARC     0x0021  /*  Speech Compression  */
#define  WAVE_FORMAT_DSPGROUP_TRUESPEECH        0x0022  /*  DSP Group, Inc  */
#define  WAVE_FORMAT_ECHOSC1    0x0023  /*  Echo Speech Corporation  */
#define  WAVE_FORMAT_AUDIOFILE_AF36     0x0024  /*    */
#define  WAVE_FORMAT_APTX       0x0025  /*  Audio Processing Technology  */
#define  WAVE_FORMAT_AUDIOFILE_AF10     0x0026  /*    */
#define  WAVE_FORMAT_DOLBY_AC2  0x0030  /*  Dolby Laboratories  */
#define  WAVE_FORMAT_GSM610     0x0031  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_MSNAUDIO   0x0032  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_ANTEX_ADPCME       0x0033  /*  Antex Electronics Corporation  */
#define  WAVE_FORMAT_CONTROL_RES_VQLPC  0x0034  /*  Control Resources Limited  */
#define  WAVE_FORMAT_DIGIREAL   0x0035  /*  DSP Solutions, Inc.  */
#define  WAVE_FORMAT_DIGIADPCM  0x0036  /*  DSP Solutions, Inc.  */
#define  WAVE_FORMAT_CONTROL_RES_CR10   0x0037  /*  Control Resources Limited  */
#define  WAVE_FORMAT_NMS_VBXADPCM       0x0038  /*  Natural MicroSystems  */
#define  WAVE_FORMAT_CS_IMAADPCM 0x0039 /* Crystal Semiconductor IMA ADPCM */
#define  WAVE_FORMAT_ECHOSC3     0x003A /* Echo Speech Corporation */
#define  WAVE_FORMAT_ROCKWELL_ADPCM     0x003B  /* Rockwell International */
#define  WAVE_FORMAT_ROCKWELL_DIGITALK  0x003C  /* Rockwell International */
#define  WAVE_FORMAT_XEBEC      0x003D  /* Xebec Multimedia Solutions Limited */
#define  WAVE_FORMAT_G721_ADPCM 0x0040  /*  Antex Electronics Corporation  */
#define  WAVE_FORMAT_G728_CELP  0x0041  /*  Antex Electronics Corporation  */
#define  WAVE_FORMAT_MPEG       0x0050  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_MPEGLAYER3 0x0055  /*  ISO/MPEG Layer3 Format Tag */
#define  WAVE_FORMAT_CIRRUS     0x0060  /*  Cirrus Logic  */
#define  WAVE_FORMAT_ESPCM      0x0061  /*  ESS Technology  */
#define  WAVE_FORMAT_VOXWARE    0x0062  /*  Voxware Inc  */
#define  WAVEFORMAT_CANOPUS_ATRAC       0x0063  /*  Canopus, co., Ltd.  */
#define  WAVE_FORMAT_G726_ADPCM 0x0064  /*  APICOM  */
#define  WAVE_FORMAT_G722_ADPCM 0x0065  /*  APICOM      */
#define  WAVE_FORMAT_DSAT       0x0066  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_DSAT_DISPLAY       0x0067  /*  Microsoft Corporation  */
#define  WAVE_FORMAT_SOFTSOUND  0x0080  /*  Softsound, Ltd.      */
#define  WAVE_FORMAT_RHETOREX_ADPCM     0x0100  /*  Rhetorex Inc  */
#define  WAVE_FORMAT_CREATIVE_ADPCM     0x0200  /*  Creative Labs, Inc  */
#define  WAVE_FORMAT_CREATIVE_FASTSPEECH8       0x0202  /*  Creative Labs, Inc  */
#define  WAVE_FORMAT_CREATIVE_FASTSPEECH10      0x0203  /*  Creative Labs, Inc  */
#define  WAVE_FORMAT_QUARTERDECK 0x0220 /*  Quarterdeck Corporation  */
#define  WAVE_FORMAT_FM_TOWNS_SND       0x0300  /*  Fujitsu Corp.  */
#define  WAVE_FORMAT_BTV_DIGITAL        0x0400  /*  Brooktree Corporation  */
#define  WAVE_FORMAT_OLIGSM     0x1000  /*  Ing C. Olivetti & C., S.p.A.  */
#define  WAVE_FORMAT_OLIADPCM   0x1001  /*  Ing C. Olivetti & C., S.p.A.  */
#define  WAVE_FORMAT_OLICELP    0x1002  /*  Ing C. Olivetti & C., S.p.A.  */
#define  WAVE_FORMAT_OLISBC     0x1003  /*  Ing C. Olivetti & C., S.p.A.  */
#define  WAVE_FORMAT_OLIOPR     0x1004  /*  Ing C. Olivetti & C., S.p.A.  */
#define  WAVE_FORMAT_LH_CODEC   0x1100  /*  Lernout & Hauspie  */
#define  WAVE_FORMAT_NORRIS     0x1400  /*  Norris Communications, Inc.  */
```

Le chunk "sdsc" pour fichiers SESANE (EVA, DIANA)

Utilisée pour la description d'un signal :

```
typedef struct
{
  WORD32 size;          // struct size
  WORD32 acronym;       // parameter's acronym ( FOURCC )
  char   paramname[80]; // parameter's name
  char   unitname[16];  // parameter's unit name
  WORD32 nsamples;      // number of samples in 'data' chunk
  WORD32 freq;          // acquisition frequency
  INT16  max;           // max value of the signal
  INT16  min;           // min value of the signal
  INT16  cmax;          // calibration at max
  INT16  czero;         // calibration at zero
  INT32  imax;          // integer part of the value at maximum
  WORD32 fmax;          // floating part x 10^6 of the maximum
} SIG_DESC;  // channel description
```

Résolution en amplitude

cmax et czero permettent de connaître la résolution:

cmax	czero	Résolution
32767	0	16 bits signés
4095	2047	12 bits non signés

Calibrage

unitname contient l'unité du signal (ex : dm3/s)
imax et fmax permettent de connaître le calibrage.

Mesure (en unitname) au max de l'échelle = imax + fmax / 10^6

imax	fmax	unitname	calibrage
20	0	hPa	20 hPa
0	200000	dm3/s	0.2 dm3/s

Comment obtenir les valeurs calibrées (ex: débit, pression, intensité SPL…) ?

Lire chunk 'sdsc' et en déduire les constantes du signal :

```
m_fSignalDynamic = (double) m_SigDesc.cmax - (double)m_SigDesc.czero;
m_fValueAtMax = (double) (m_SigDesc.imax + m_SigDesc.fmax / 1000000.0);
```

Pour une valeur d'échantillon lu et codé sur un INT16, on obtient la mesure calibrée:

```
double val_calib = (double)(valint16 - czero) * (m_fValueAtMax / m_fSignalDynamic)
```

Exemple : signal de pression, `cmax = 32767, czero = 0, imax = 20, fmax = 0`
`fSignalDynamic = 32767, fValueAtMax = 20`

valint16 [-32768; + 32767]	val_calib (en hPa)
0	0
32767	20.0
1000	0.610
-5325	-3.25

Liste des acronymes des données SESANE

L'acronyme est une variable FOURCC qui identifie de façon non ambiguë le type de données : débit oral, pression, signal sonore…

```
#define ID_OAF  MK_FOURCC('o','a','f',' ') // Oral Air Flow
#define ID_NAF  MK_FOURCC('n','a','f',' ') // Nasal Air Flow
#define ID_F0   MK_FOURCC('f','0',' ',' ') // Fundamental frequency
#define ID_F02  MK_FOURCC('f','0','2',' ') // Fundamental frequency on second channel
#define ID_GLO  MK_FOURCC('g','l','o',' ') // Glottographic signal
#define ID_IOP  MK_FOURCC('i','o','p',' ') // Intra-oral pressure
#define ID_SGP  MK_FOURCC('s','g','p',' ') // Sub-glottal pressure
#define ID_WAV  MK_FOURCC('w','a','v','e') // Speech signal
#define ID_WA2  MK_FOURCC('w','a','v','2') // Speech signal
#define ID_WA3  MK_FOURCC('w','a','v','3') // Speech signal
#define ID_WA4  MK_FOURCC('w','a','v','4') // Speech signal
#define ID_WF0  MK_FOURCC('w','f','0',' ') // Filtered speech signal for f0 computation
#define ID_MYO  MK_FOURCC('m','y','o',' ') // Electromyographic signal (aux input)
#define ID_MY1  MK_FOURCC('m','y','o','1') // Electromyographic signal
#define ID_MY2  MK_FOURCC('m','y','o','2') // Electromyographic signal
#define ID_MY3  MK_FOURCC('m','y','o','3') // Electromyographic signal
#define ID_MY4  MK_FOURCC('m','y','o','4') // Electromyographic signal
#define ID_MY5  MK_FOURCC('m','y','o','5') // Electromyographic signal
#define ID_MY6  MK_FOURCC('m','y','o','6') // Electromyographic signal
#define ID_TIM  MK_FOURCC('t','i','m','e') // Timing
#define ID_FFT  MK_FOURCC('f','f','t',' ') // Fast Fourier Transform
#define ID_INT  MK_FOURCC('i','n','t',' ') // Intensity
#define ID_INT2 MK_FOURCC('i','n','t','2') // Intensity on second channel
#define ID_SF   MK_FOURCC('s','f',' ',' ') // Singing formant
#define ID_SPC  MK_FOURCC('s','p','e','c') // Spectogram
#define ID_EPG  MK_FOURCC('e','p','g',' ') // Electropalatographic signal
#define ID_MVX  MK_FOURCC('m','o','v','x') // x axis movement signal
#define ID_MVY  MK_FOURCC('m','o','v','y') // y axis movement signal
```

Le chunk "adsc" pour fichiers SESANE (EVA, DIANA)

Utilisée pour la description de l'acquisition :

typedef struct

```
{
  WORD32 size;     // struct size
  WORD16 nch;      // number of channels
  WORD32 nsamples; // number of samples
  WORD32 freq;     // acquisition frequency
  WORD16 bps;      // bits per sample
  INT32  highest;  // highest value
  INT32  lowest;   // lowest value
  INT32  zero;     // zero
  WORD16 reccode;  // recording program code
  WORD16 recver;   // version of the acquisition program

  // code   software    version
  // 0      Unknown        -
  // 1      Physiologia    0       linearized oral airflow
  // 2      Eva            0       linearized oral airflow
  // 3      Diana          0
  // 4      EVA 2          0       linearized oral airflow

} ACQ_DESC;      // Acquisition Description
```

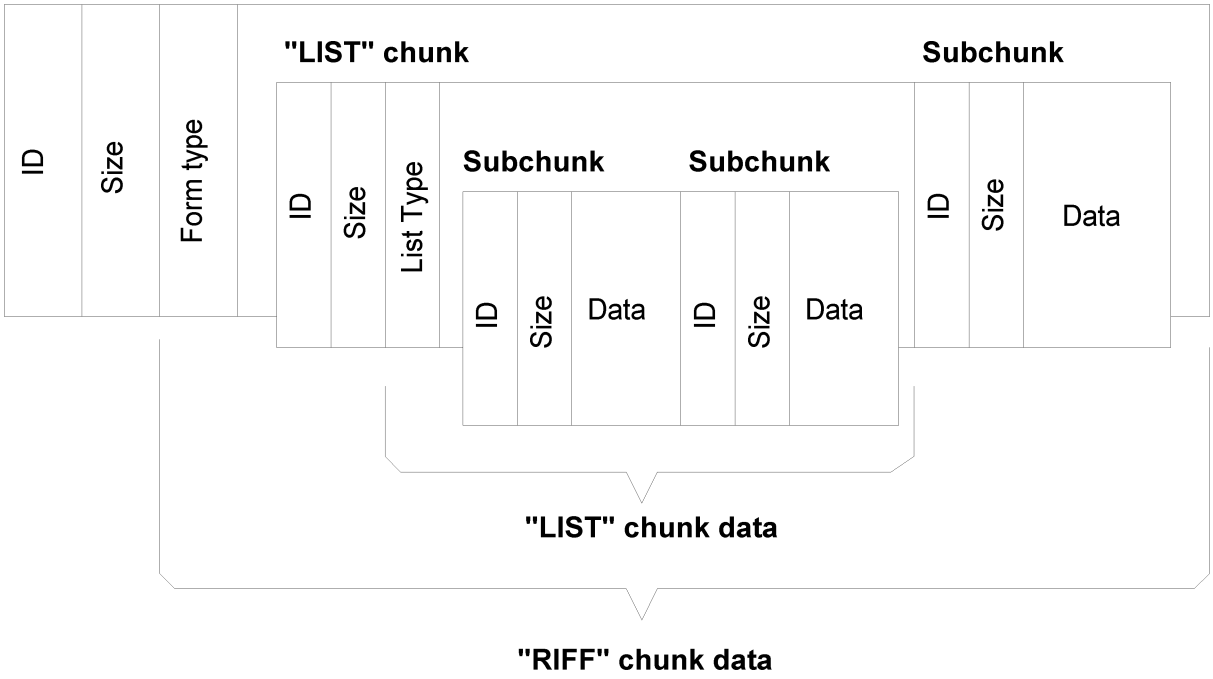
Le chunk "LIST"

Le chunk "LIST" permet de rajouter des informations supplémentaires. Il incorporent lui aussi un champ FOURCC en début de zone de données pour coder le format de type de liste.
Par exemple, un chunk "LIST" avec un listtype = "INFO" peut contenir les sous-chunks "ICOP" et "ICRD" qui renseignent sur le copyright et l'information de création

```
/*////////////////////////////////////////////////////////////////////////// */

/*              INFO LIST CHUNKS (from the Multimedia Programmer's Reference
                                      plus new ones)
*/
#define RIFFINFO_IARL      mmioFOURCC ('I', 'A', 'R', 'L')     /*Archival location  */
#define RIFFINFO_IART      mmioFOURCC ('I', 'A', 'R', 'T')     /*Artist  */
#define RIFFINFO_ICMS      mmioFOURCC ('I', 'C', 'M', 'S')     /*Commissioned  */
#define RIFFINFO_ICMT      mmioFOURCC ('I', 'C', 'M', 'T')     /*Comments  */
#define RIFFINFO_ICOP      mmioFOURCC ('I', 'C', 'O', 'P')     /*Copyright  */
#define RIFFINFO_ICRD      mmioFOURCC ('I', 'C', 'R', 'D')     /*Creation date of subject  */
#define RIFFINFO_ICRP      mmioFOURCC ('I', 'C', 'R', 'P')     /*Cropped  */
#define RIFFINFO_IDIM      mmioFOURCC ('I', 'D', 'I', 'M')     /*Dimensions  */
#define RIFFINFO_IDPI      mmioFOURCC ('I', 'D', 'P', 'I')     /*Dots per inch  */
#define RIFFINFO_IENG      mmioFOURCC ('I', 'E', 'N', 'G')     /*Engineer  */
#define RIFFINFO_IGNR      mmioFOURCC ('I', 'G', 'N', 'R')     /*Genre  */
#define RIFFINFO_IKEY      mmioFOURCC ('I', 'K', 'E', 'Y')     /*Keywords  */
#define RIFFINFO_ILGT      mmioFOURCC ('I', 'L', 'G', 'T')     /*Lightness settings  */
#define RIFFINFO_IMED      mmioFOURCC ('I', 'M', 'E', 'D')     /*Medium  */
#define RIFFINFO_INAM      mmioFOURCC ('I', 'N', 'A', 'M')     /*Name of subject  */
#define RIFFINFO_IPLT      mmioFOURCC ('I', 'P', 'L', 'T')     /*Palette Settings. No. of colors requested.   */
#define RIFFINFO_IPRD      mmioFOURCC ('I', 'P', 'R', 'D')     /*Product  */
#define RIFFINFO_ISBJ      mmioFOURCC ('I', 'S', 'B', 'J')     /*Subject description  */
#define RIFFINFO_ISFT      mmioFOURCC ('I', 'S', 'F', 'T')     /*Software. Name of package used to create file.  */
#define RIFFINFO_ISHP      mmioFOURCC ('I', 'S', 'H', 'P')     /*Sharpness.  */
#define RIFFINFO_ISRC      mmioFOURCC ('I', 'S', 'R', 'C')     /*Source.  */
#define RIFFINFO_ISRF      mmioFOURCC ('I', 'S', 'R', 'F')     /*Source Form. ie slide, paper  */
#define RIFFINFO_ITCH      mmioFOURCC ('I', 'T', 'C', 'H')     /*Technician who digitized the subject.  */
```

"RIFF" Chunk



Le chunk "INFO" : structure FILE_DESC (pour fichiers WSIG)

La structure FILE est utilisée pour stocker des commentaires, la date de création, le contenu... des enregistrements.

```
typedef struct
{
  WORD32 size;                // struct size
  char   creationdate[20];    // format yyyy/mm/dd
  char   copyright[80];       // when apropriate
  char   recorder[80];        // recording software name
  char   documentname[80];    // contains the sentence
  char   documentcomment[512]; // experimental comment

} FILE_DESC;
```

Structure obsolète.

Le chunk LIST | INFO est composé des sous-chunks liés à la structure FILE_DESC de la façon suivante :

ID_INAM	FileDesc.documentname
ID_ICMT	FileDesc.documentcomment
ID_ICRD	FileDesc.creationdate
ID_ICOP	FileDesc.copyright
ID_ISFT	FileDesc.recorder

Exemple d'entête de fichier SESANE (RIFF-WSIG)

```
RIFF  [81231]
      WSIG
      sdsc  [128]
            => acronym =      543580527   (oaf )
            => paramname =    oral airflow
            => unitname =     dm3/s
            => nsamples =     40448
            => freq =         6250
            => max =          10665
            => min =          -3395
            => cmax =         32767
            => czero =        0
            => imax =         2
            => fmax =         0
      adsc  [32]
            => nch =          16
            => nsamples =     0
            => freq =         100000
            => bps =          16
            => highest =      32767
            => lowest =       -32768
            => zero =         0
            => reccode =      4     (EVA2)
            => recver =       0
      LIST  [139]
            INFO
            INAM  [4]
                  => documentname = P+A
            ICMT  [29]
                  => documentcomment =    [Test][1][0][0][P+A][][]
            ICRD  [14]
                  => creationdate = 1999-7-27
            ICOP  [19]
                  => copyright =    (c) SQLab 1998
            ISFT  [29]
                  => recorder =     Glottal Efficiency Index
      data  [80896]
```

Résumé sur la structure d'un fichier SESANE

Pour plus de détails, voir paragraphes précédents décrivant chaque Chunk.

Adresse (en octets)	Donnée	Taille donnée	Commentaire
000	'R' 'I' 'F' 'F'	FOURCC	Un fichier RIFF commence systématiquement par les 4 caractères 'R' 'I' 'F' 'F'
004	Size	DWORD	Taille du chunk RIFF
008	Type Riff	FOURCC	'W' 'S' 'I' 'G' pour fichiers SESANE[2]
012	ID chunk Signal Descripteur	FOURCC	's' 'd' 's' 'c' pour fichiers SESANE[3]
016	Size	DWORD	Taille du chunk SigDesc = 128 octets
020	size;	DWORD	Taille de la structure = 128octets
024	acronym;	DWORD	parameter's acronym
028	paramname;	char[80]	parameter's name
108	Unitname;	char[16]	parameter's unit name
124	nsamples	DWORD	number of samples in 'data'
128	Freq ;	DWORD	Sampling Rate
132	max;	INT16	max value of the signal
134	min;	INT16	min value of the signal
136	cmax;	INT16	calibration at max
138	czero;	INT16	calibration at zero
140	imax;	INT32	integer part of the value at maximum
144	fmax;	DWORD	floating part x 10^6 of the maximum
148	ID chunk Acq Descripteur	FOURCC	'a' 'd' 's' 'c' pour fichiers SESANE[4]
152	Size	DWORD	Taille du chunk AcqDesc = 32 octets
156	size;	DWORD	Taille de la structure = 32 octets
160	nch; //	WORD	number of channels
162	nsamples; //	DWORD	number of samples
166	freq; //	DWORD	acquisition frequency
170	bps; //	WORD	bits per sample
172	highest; //	INT32	highest value
176	lowest; //	INT32	lowest value
180	zero; //	INT32	zero
184	reccode; //	WORD	recording program code
186	recver; //	WORD	version of the acquisition program
188	'L' 'I' 'S' 'T'	FOURCC	Chunk LIST
192	Size	DWORD	Taille du chunk LIST : variable selon taille (ex: 153)
196	'I' 'N' 'F' 'O'	FOURCC	Chunk LIST – INFO
192	Size	DWORD	Taille du chunk LIST : variable
200	Subchunk Info 1	FOURCC	Ex: 'INAM' ⇔ Name
204	Size Subchunk 1	DWORD	Ex: 19
208-226	String 1	char[sz1]	Ex: "lec chèvre normale"
227	Subchunk Info 2	FOURCC	Ex: 'ICMT' ⇔ Commentaires
231	Size Subchunk 2	DWORD	Ex: 49
235-283	String 2	char[sz2]	Ex: [Gaston][Michèle][2][63][chèvre normale][][]
284	Subchunk Info 3	FOURCC	Ex: 'ICRD' ⇔ Creation date
288	Size Subchunk 3	DWORD	Ex : 14
292-305	String 3		Ex: 2006-9-26
306	Subchunk Info 4	FOURCC	Ex: 'ICOP' ⇔ Copyright
310	Size Subchunk 4	DWORD	Ex : 19
314-332	String 4		Ex: (c) SQLab 1998
333	Subchunk Info 5	FOURCC	Ex: 'ISFT' ⇔ Software
337	Size Subchunk 5	DWORD	Ex : 8
341-348	String 5		Ex: Prosody
349[5]	'd' 'a' 't' 'a'	FOURCC	Chunk DATA
353	Size data	DWORD	Ex : 3747840
0000357	**Sample[0]**	**INT16**	**Echantillon sur 2 octets (16 bits signés)**
0000359	**Sample[1]**	**INT16**	
0000361	**Sample[2]**	**INT16**	
…3748195	**Sample[n-1]**	**INT16**	
3748197			

[2] 'W' 'A' 'V' 'E' pour fichier .wav
[3] 'f' 'm' 't' ' ' pour fichier .wav
[4] N'existe pas pour fichier Wave
[5] Cette valeur varie à chaque fichier. Elle se calcule ainsi : adresse_data=196(adresse chunk INFO) + 153 (taille Chunk LIST)

Procédure simple de lecture des fichiers SESANE (EVA, DIANA)

1. (Lire fichier à adresse 0 : vérifier présence des 4 caractères `'R' 'I' 'F' 'F')`
2. (Lire fichier à adresse 8 : vérifier présence des 4 caractères `'W' 'S' 'I' 'G')`

3. Lire fichier à adresse 12 : vérifier présence des 4 caractères `'s' 'd' 's' 'c'`
4. Si OK, lire fichier à adresse 128 => fréquence d'échantillonnage codée sur un DWORD

5. Lire fichier à adresse 188 : vérifier présence des 4 caractères `'L' 'I' 'S' 'T'`
6. Si OK, lire fichier à adresse 192 => 'chunksize' (codée sur un DWORD) du chunk LIST-INFO textuel
7. Sauter 'chunksize' octets textuels (ex: 153 avec exemple précédent)

8. Lire fichier à adresse 188 + 4 + 4 + 'chunksize' (ex :188+4+4+153=349) :
   vérifier présence des 4 caractères `'d' 'a' 't' 'a'`

9. Lire fichier à adresse suivante : taille de la masse de données en octets codée sur un DWORD, le nb d'échantillons (INT16 ⇔ 2 octets) est la moitié de la valeur trouvée

10. Lire le flot d'échantillons en INT16 (short int)

Alternative à étapes 5,6,7,8 : chercher de façon itérative 4 caractères consécutifs `'d' 'a' 't' 'a'` qui signalent le bloc de données. Attention, passer par étape 9 avant étapes 10.